Software Overview

Year: 2023 Semester: Fall Team: 16 Project: Air Hockey Playing Robot

Creation Date: 09/07/2023 Last Modified: September 9, 2023

Author: Joey Collins Email: colli384@purdue.edu

Assignment Evaluation: See Rubric on Brightspace Assignment

1.0 Software Overview

The main software components of the Air Hockey Playing Robot are as follows:

* [PC] Analyze video frames for specified objects
* [PC] Formulate predictions for future puck locations
* [Microcontroller] Read in sensor data
* [PC] Serialize data for communication between PC and microcontroller
* [Microcontroller] Establish communication between microcontroller and motor control module
  1. *Analyze Video Frames for Specified Objects*

The majority of the software development on this project will be focused on video analysis. The camera stationed several feet above the center of the table will provide an image containing both the puck and the robot mallet. The camera will be capturing images at 120 frames per second, which will be immediately sent to the PC running the image processing software. The processing will be done using the OpenCV library for Python. The specifics will be discussed in the Description of Algorithms section.

* 1. *Formulate Predictions for Future Puck Locations*

The computation being done to predict the trajectory of the puck is being done on the PC. The previous puck locations will be used to determine the path that the puck is currently on and where the robot mallet should move to in order to intercept the puck. A more detailed description will be located in the Description of Algorithms section.

* 1. *Read in Sensor Data*

The sensors being used in this project include the camera used for live video and the photodiode used for score detection. The team is not focusing on the camera as a sensor, as there is no design effort necessary to acquire the images from the camera. However, the photodiode does require design effort. The PD333-3B/H0/L2 photodiode will output an analog signal that will be converted into a digital signal by the ADC on the microcontroller. Whenever the digital signal crosses a certain threshold, an interrupt will occur, telling the microcontroller to add a goal to the corresponding team’s score.

* 1. *Serialize Data for Communication between PC and Microcontroller*

The data being sent from the PC to the microcontroller will include information about the current mallet position and the location that the motors should move the mallet towards by calculating the displacement necessary to reach the desired location. This information will be sent out in standard serialized packets. Each packet will be created and sent after the processing of an individual frame, allowing for small adjustments as the puck moves closer to the robot’s side of the table.

* 1. *Establish Communication between Microcontroller and Motor Control Module*

The microcontroller will convert the packets sent by the PC into instructions for the motor control modules. There will be two of these modules, one for each of the motors. The microcontroller will decode the desired displacement vectors into two separate instructions to send to the motor control modules.

A combination of efficient code and a focus on time complexity is important for this project. The successful implementation of these key components will allow for a unique and responsive system.

2.0 Description of Algorithms

*2.1 Video Frame Analysis*

As stated previously, the video frames are being processed by OpenCV algorithms. The first step in this process is converting the frame into the HSV color spectrum, as opposed to the BGR color spectrum. This makes the task of filtering certain colors more manageable. Prior to this, a color range specific to the puck and a color range specific to the robot mallet will be chosen. These ranges will be used to filter the HSV frame, leaving only the pixels that fit into those ranges. This means that the only objects that have non-zero values will be the puck and the robot mallet. Next, these two objects will be given coordinates relative to the pixel-based coordinate system. These will need to be converted to coordinates relative to the discrete grid on the table. This will be useful when giving the motors instructions on where to move the robot mallet.

*2.2 Puck Path Prediction*

The puck path prediction will make use of the coordinates given to the puck in the video analysis component. Every time a new frame is processed, an updated location is given to the puck. The displacement between the new location and the previous location will be used to determine a pseudo velocity vector where the units are displacement per frame. Before any kind of predictions can be made, the boundaries of the table in the image must be established due to the potential for bounces off of walls. With the velocity vector and prediction for wall locations, an estimated path can be created. If the vector intersects with the wall, then the angle of incidence will be equal to the angle of reflection. This information can be used to provide an estimation for the path of the puck after it bounces off of a wall. This estimation does not explicitly consider that friction along the table’s surface will cause the puck to slow down. However, because the puck location is being constantly updated, the deceleration will be actively accounted for. If the team finds that it would be more accurate to directly calculate the velocity after taking friction into account, then the formula shown below could be used [1].

A black text on a white background

Description automatically generated

*Figure 1*

The K represents the coefficient of friction on the table’s surface, the dt is time between measurements, and the m is the mass of the puck.

3.0 Description of Data Structures

There are 3 main types of data structures being used for the Air Hockey Playing Robot. The first of which is the package in which information is being sent from the PC to the microcontroller. This will be done using the UART protocol, which means that each send will include 8 bits of actual data. Currently, the data that each packet contains will be structured as follows: an instruction to move in the x-direction will have the most significant bit equal to 0. The rest of the 7 bits will be used to determine how far to move the robot mallet in the x-direction. For example, a frame containing 0b00000101 will instruct the microcontroller to move the robot mallet 5 units in the positive x-direction. The same goes for the y-direction, but the most significant bit will be set to 1. In order to allow for movement in the negative direction, the two’s complement format will be used. A frame containing 0b11111011 will tell the microcontroller to move the robot mallet 5 units in the negative y-direction.

Another type of data structure being used in this project is the matrix. Each frame being processed is a matrix containing pixel information about color. OpenCV is heavily reliant upon this data structure, which makes it vital to the video analysis component of this project.

The last major type of data structure being used is a lookup table for motor instructions. The microcontroller will have to translate the packets coming from the PC that have coordinate system information into instructions for the motor control modules. It would be computationally expensive to calculate this each time a packet is received by the microcontroller. In order to maintain time efficiency with this project, a more memory consuming approach is the correct course of action.

4.0 Sources Cited:

[1] R. Geren. “Charger Robotic Air hockey-Modified Air Hockey Table Featuring Automated Play Against a Human Opponent.” louis.uah.edu. https://louis.uah.edu/cgi/viewcontent.cgi?article=1756&context=honors-capstones (accessed 9/7/2023).

Appendix 1: Program Flowcharts

*A diagram of a software process

Description automatically generated*

PC Flowchart

*A diagram of a flowchart

Description automatically generated*

Future Puck Location Estimation Flowchart

*A diagram of a program

Description automatically generated*

Microcontroller Flowchart

Appendix 2: State Machine Diagrams

*A diagram of a computer system

Description automatically generated*

Microcontroller State Machine Diagram